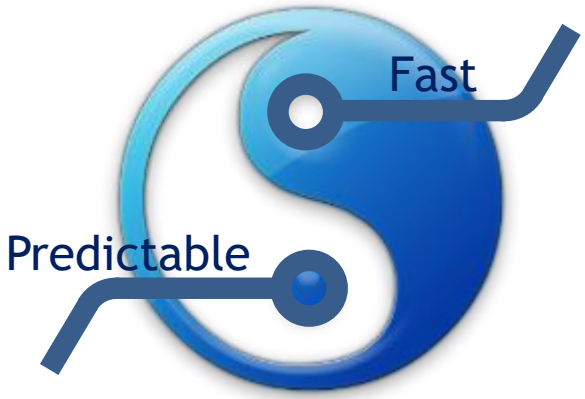
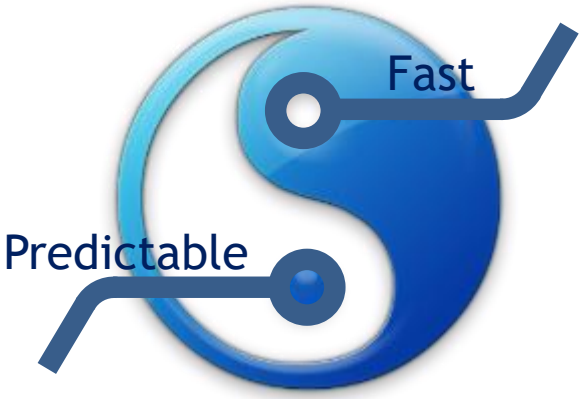
Validating static WCET analysis: a method and its application



Wei-Tsun Sun IRT Saint Exupéry
Eric Jenn IRT Saint Exupéry, second by Thales AVS.
Hugues Cassé IRIT, Université de Toulouse

09/07/2019 @ WCET 2019, Stuttgart, Germany



The Project CAPHCA

An IRT team, funded with ANR

WCET (Worst-Case Execution Time)

Why do we need to care about the worst-case?

Why WCET?

- Know the worst, still schedule-able, then schedule-able for all cases.
- No bad surprises for time-critical parts.
- Safety, and to be Sure.

How to get WCET?

- **Static approaches and measurement approaches.**
- **Add some flavours of statistics.**
- **Approaches are good, but we need to be Sure.**
- **Who performs the analysis/measurements?**
  - Human
  - Program
- **Who wrote the programs?**
  - Human
  - Compiler, code-gen
- **Who wrote the compiler……?**
  - Human……

One of the problems of being human

- **Human makes mistakes.**

- **For WCET analysis, sometimes HW info needed:**
  - Looking into device drivers so you know all the details (takes a lot of time)
  - Little bird whispering next to your ears (takes too much imagination)
  - Existing models provided by vendors (takes some luck)
  - Data-sheet (mostly accessible, detailed…….too detailed, 5000+ pages)

- **Common pitfall and possible remedy**
  - WYPMNBWYG

- **Need to find a way to detect the mistake**
  - Comparisons

Checking the WCET tool

To make sure we are on the right track

We need to make sure the estimated WCET is trust-worthy

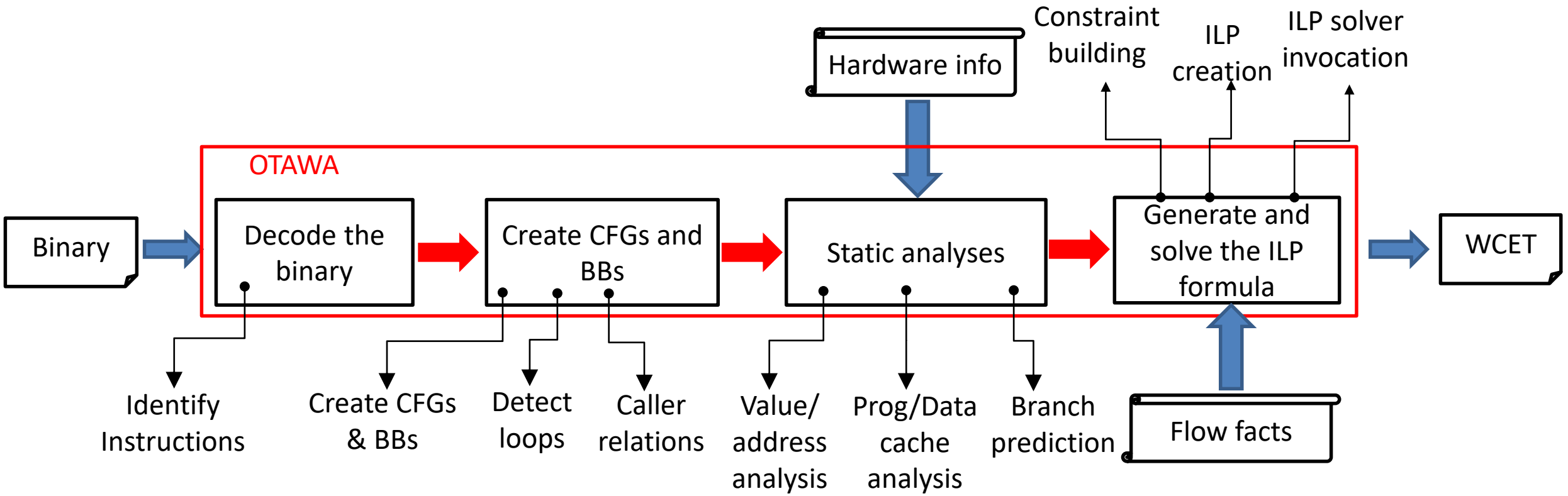
- **WCET estimation from WCET tool.**
- **Need to check WCET tool to make sure:**
  - It is constructed correctly
  - Its implementation satisfies the theory-to-implement, e.g. abstract interpretation
- **Our case:**
  - WCET tool to check – OTAWA
    - From IRIT, University of Toulouse
    - Open source
    - Support ARM, RISC-V, PowerPC, Kalray MPPA, and our interested target: Infineon TriCore
- **Infineon TriCore is one of our target because:**
  - Three CPUs
  - Targeted for automobile
  - Powerful co-processors and peripherals
  - Good candidate for REAL case-study, not just toy examples
  - Interference analysis
  - Many other tools support it, make a good case for tool-chain integration

What does OTAWA do?

- OTAWA takes the program binary as the input
- Provides WCET (in CPU cycles)

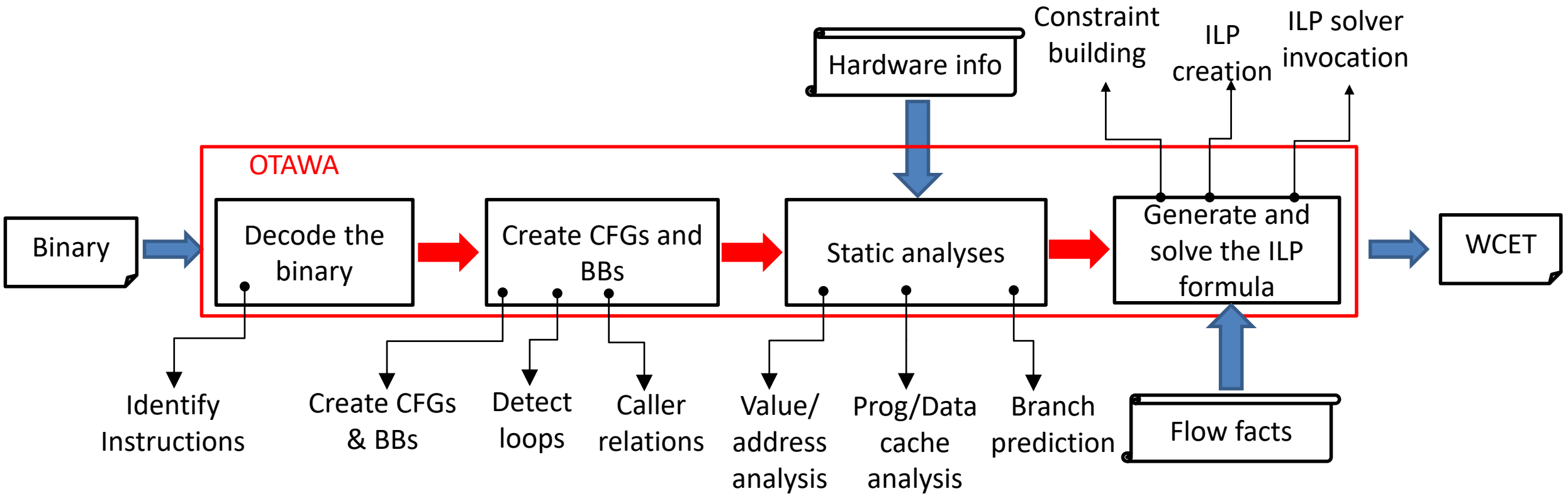
OTAWA is a huge framework of many years work

❑ OTAWA is sophisticate



❑ To support a new architecture by OTAWA

- To be able to decode the binary
- To have static analyses available (program cache analysis, data cache analysis, …)



Analyses are platform independent and can be configured

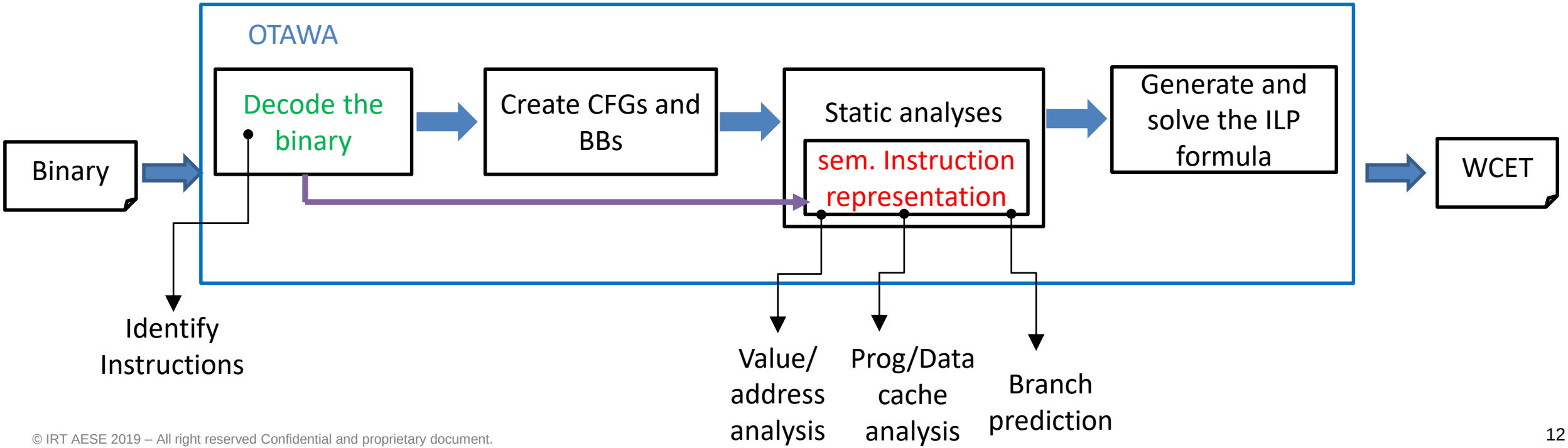
[1] Hugues Cassé, Florian Birée, and Pascal Sainrat. Multi-architecture value analysis for machine code. In 13th International Workshop on Worst-Case Execution Time Analysis, pages pp–42, 2013.

❑ Analyses in OTAWA are made

- Platform independent, so developer can focus on the analysis itself
- Each instruction is presented by a sequence of *semantic instructions* [1]
  - e.g.

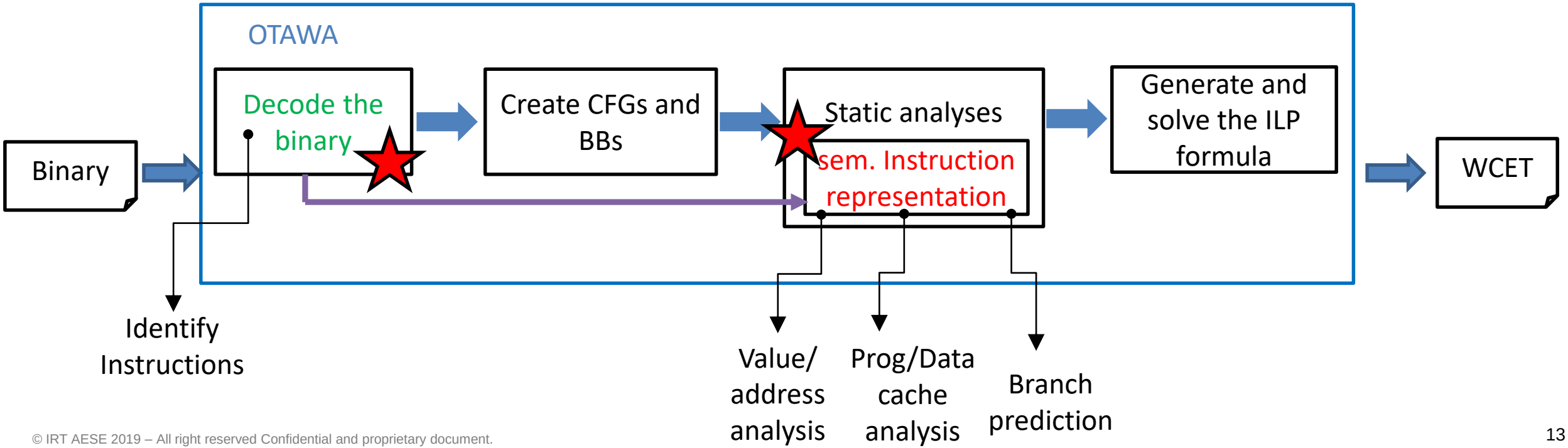
add r1, r2, 3 → set t1, 3
add r1, r2, t1

❑ This is done before analyses



Analyses are platform independent and can be configured

- **We want to check**
  - The binary decoding and semantic instructions
  - The static analyses



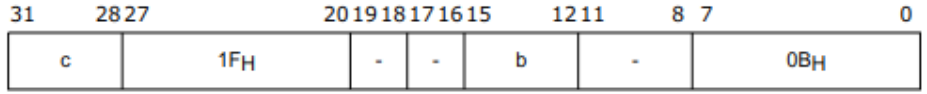
Validation of the ISA model for the instruction decoding

The binary decoding

- The ISA is described in NMP format [2]
- Normally processor has user-manual for ISA [3]

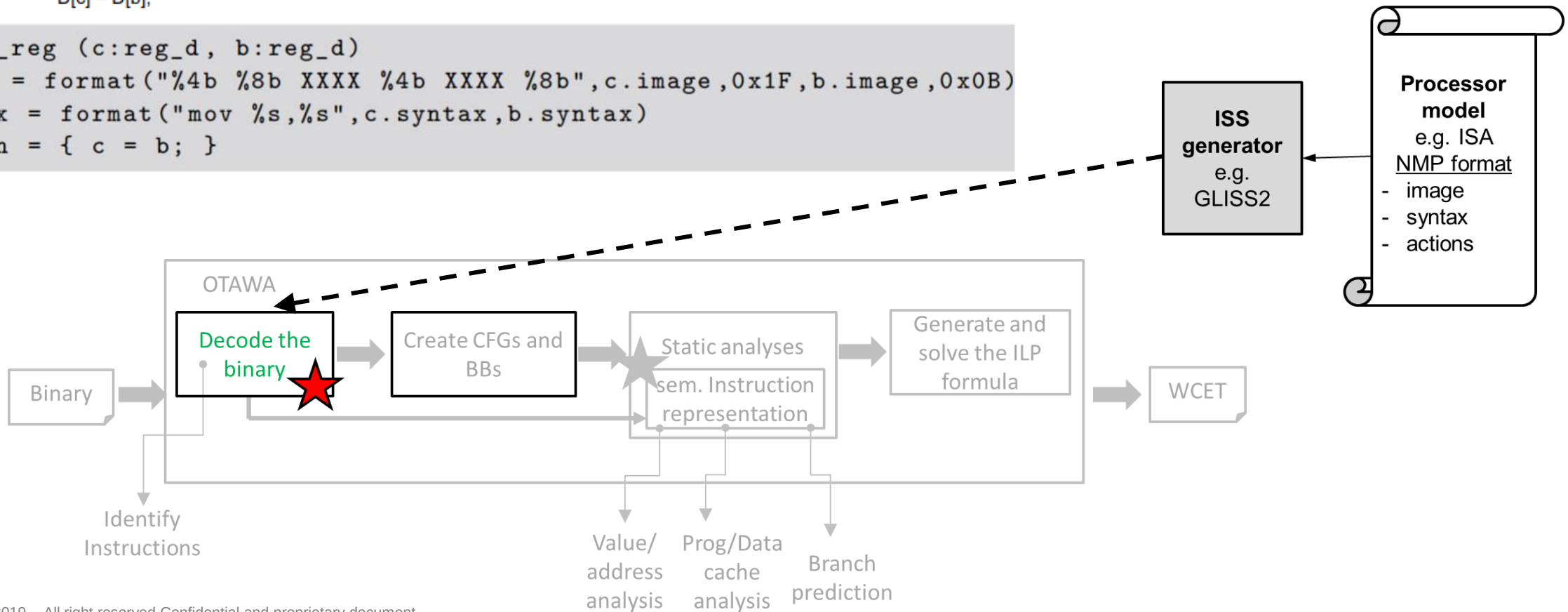
[2] Johan Van Praet, Dirk Lanneer, Werner Geurts, and Gert Goossens. nml: A structural processor modeling language for retargetable compilation and asip design. In Processor Description Languages, pages 65–93. Elsevier, 2008.
[3] TriCore™ TriCore™ V1.6 Microcontrollers User Manual (Volume 2).

MOVD[c], D[b] (RR)



D[c] = D[b];

```
op mov_reg (c:reg_d, b:reg_d)
 image = format("%4b %8b XXXX %4b XXXX %8b",c.image,0x1F,b.image,0x0B)
 syntax = format("mov %s,%s",c.syntax,b.syntax)
 action = { c = b; }
```



The binary decoding

- The ISA is described in NMP format [2]
- Normally processor has user-manual for ISA [3]

[2] Johan Van Praet, Dirk Lanneer, Werner Geurts, and Gert Goossens. nml: A structural processor modeling language for retargetable compilation and asip design. In Processor Description Languages, pages 65–93. Elsevier, 2008.
[3] TriCore™ TriCore™ V1.6 Microcontrollers User Manual (Volume 2).

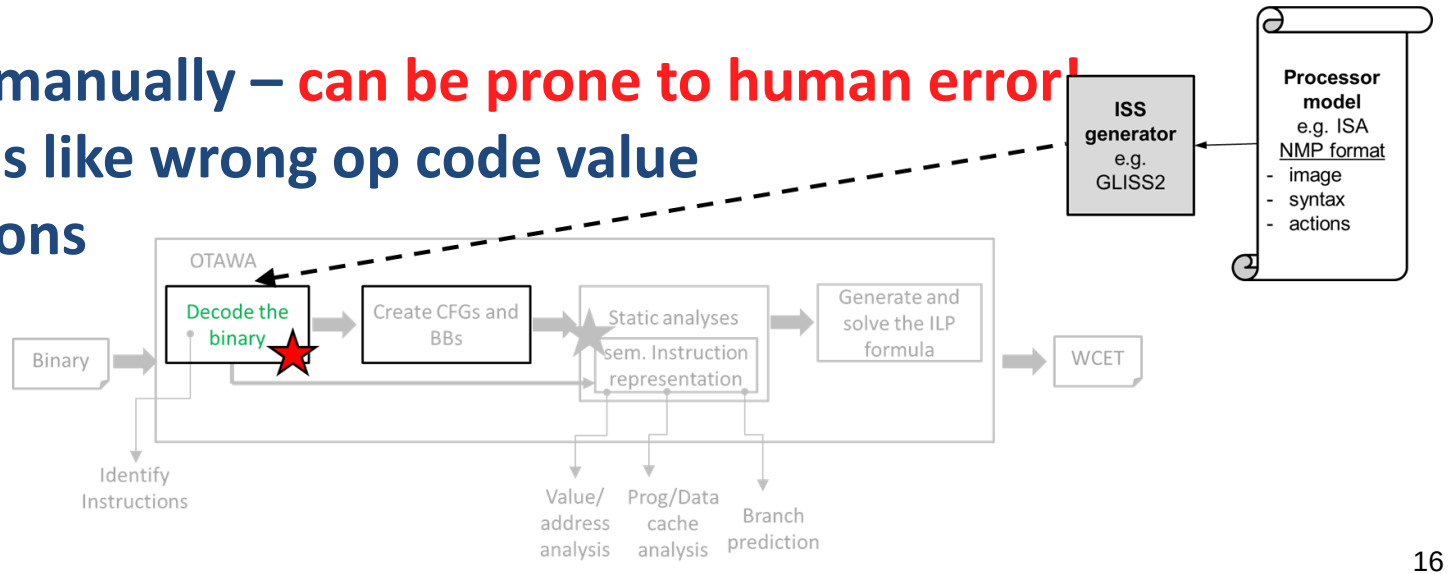
MOVD[c], D[b] (RR)

31	28	27	20	19	18	17	16	15	12	11	8	7	0
c		$1F_H$		-	-	b		-		$0B_H$			

D[c] = D[b];

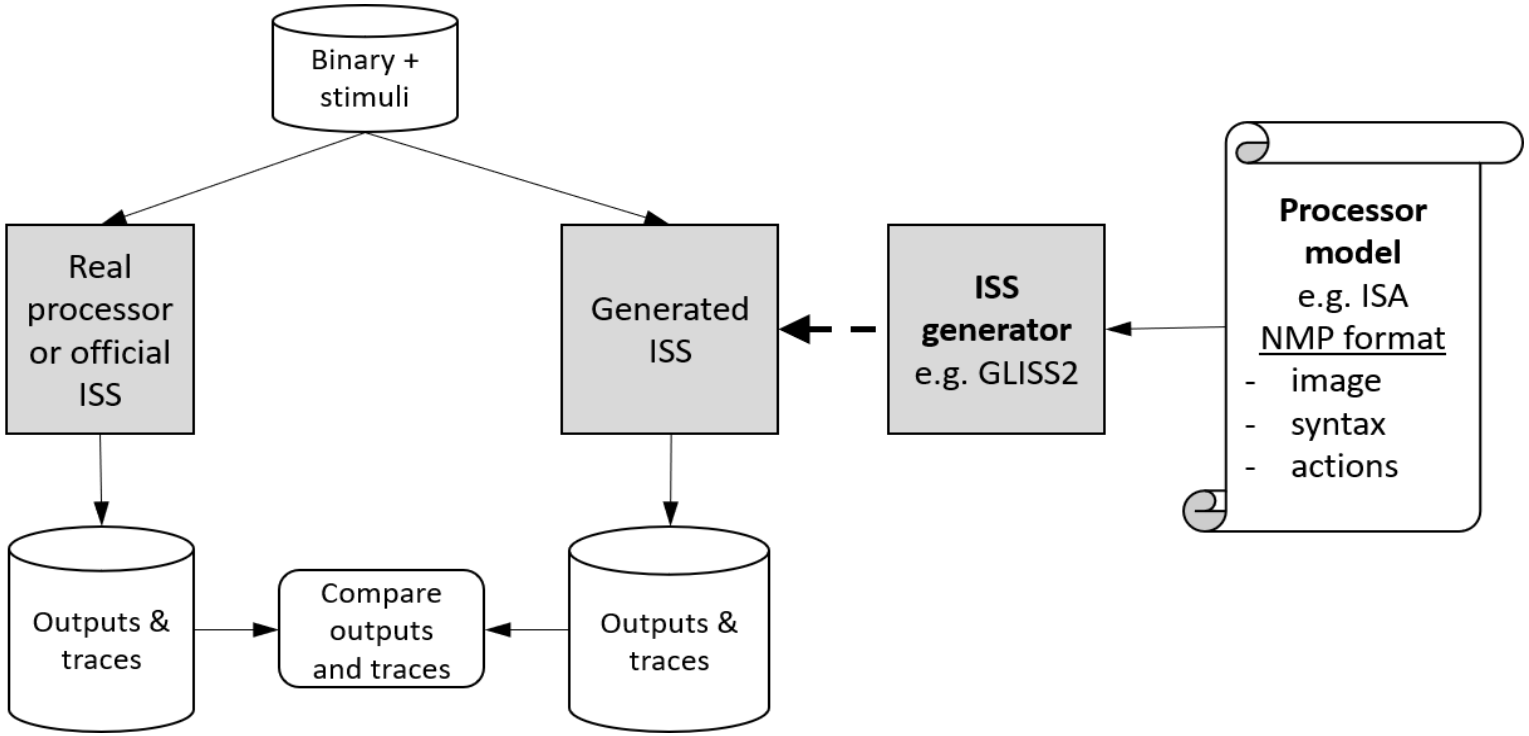
```
op mov_reg (c:reg_d, b:reg_d)
 image = format("%4b %8b XXXX %4b XXXX %8b",c.image,0x1F,b.image,0x0B)
 syntax = format("mov %s,%s",c.syntax,b.syntax)
 action = { c = b; }
```

- The description of ISA is captured manually – can be prone to human error!
- The GLISS2 can not found problems like wrong op code value
- 815 OPCODE-OPERAND combinations



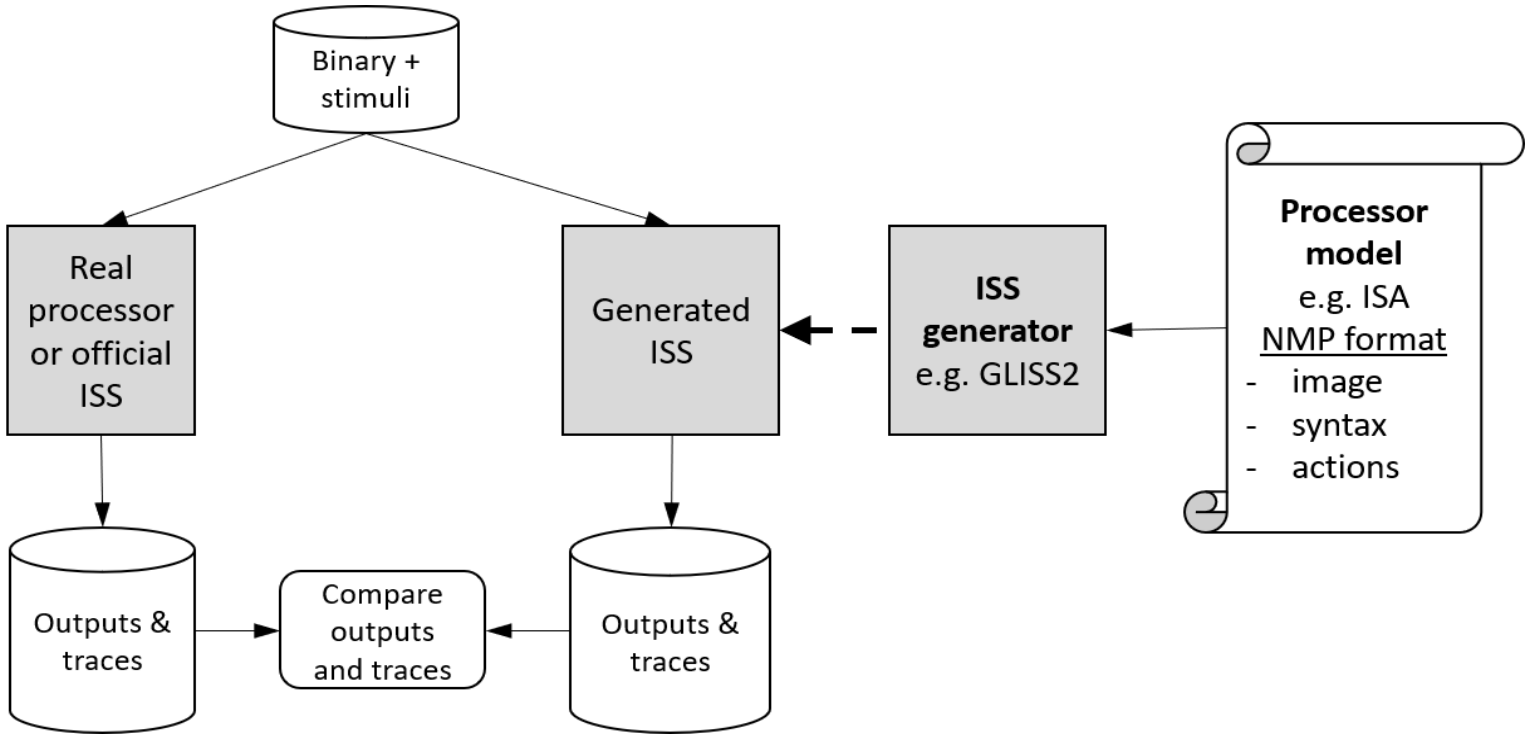
Checking the instruction decoding / interpretation

- **Official ISS from Infineon: TSIM**
- **OTAWA is able to generated ISS from NMP files**
  - Describing ISA
  - Use to decode instructions in OTAWA
  - First step to support a new architecture
- **Compare the "processor state" between TSIM and OTAWA-ISS**
  - Register values
  - Memory accesses



Some results

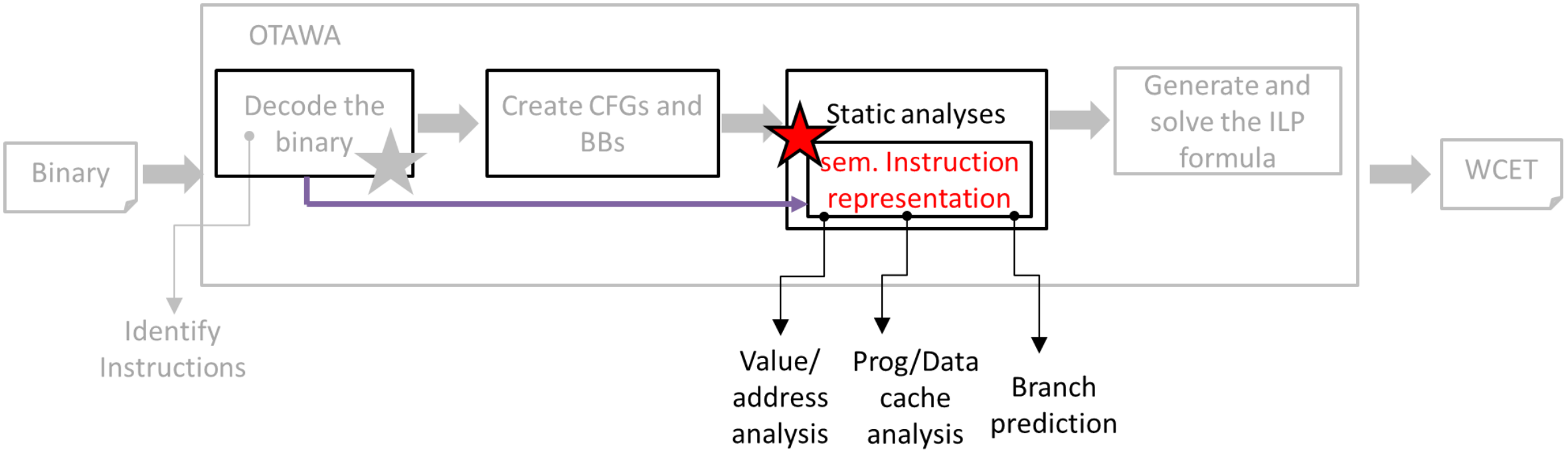
- **815 combinations of Instructions + operands**
  - 203 (25%) were covered.
  - ~200,000,000 instructions were executed in 36 applications including twIRTee and dual-eMotor use cases.
  - ~20% of OTAWA's TriCore instruction of decoding were corrected.
  - Mostly human error when writing the NMP file.



Validation of static analyses and semantic instruction mappings

Validate the static analyses

- Similarly we check for other analyses in OTAWA
- For example abstract interpretation in value analysis



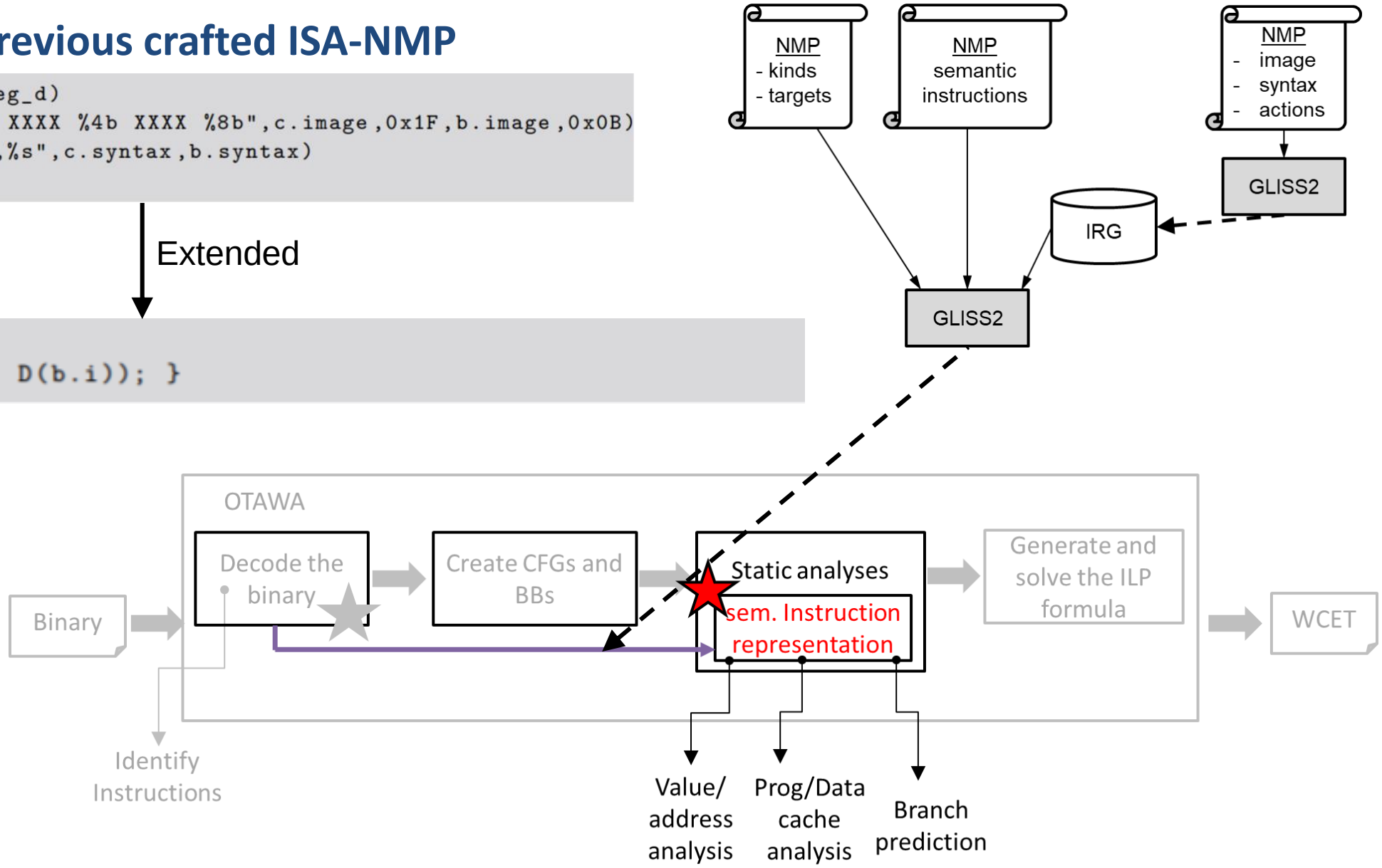
Capturing the semantics of the instructions

❑ Extended the previous crafted ISA-NMP

```
op mov_reg (c:reg_d, b:reg_d)
 image = format("%4b %8b XXXX %4b XXXX %8b",c.image,0x1F,b.image,0x0B)
 syntax = format("mov %s,%s",c.syntax,b.syntax)
 action = { c = b; }
```

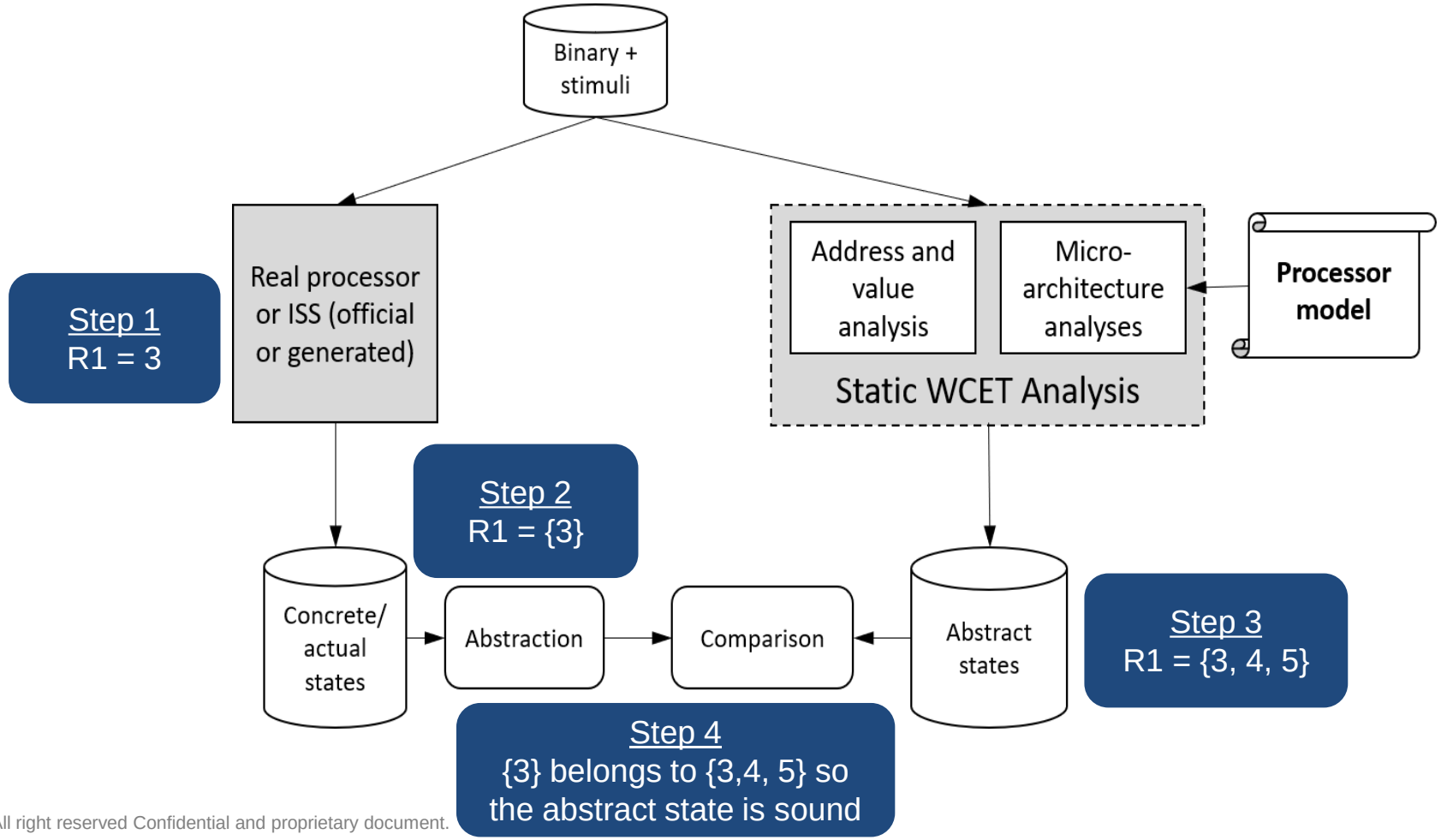
Extended

```
extend mov_reg
  sem = { SET(D(c.i), D(b.i)); }
```



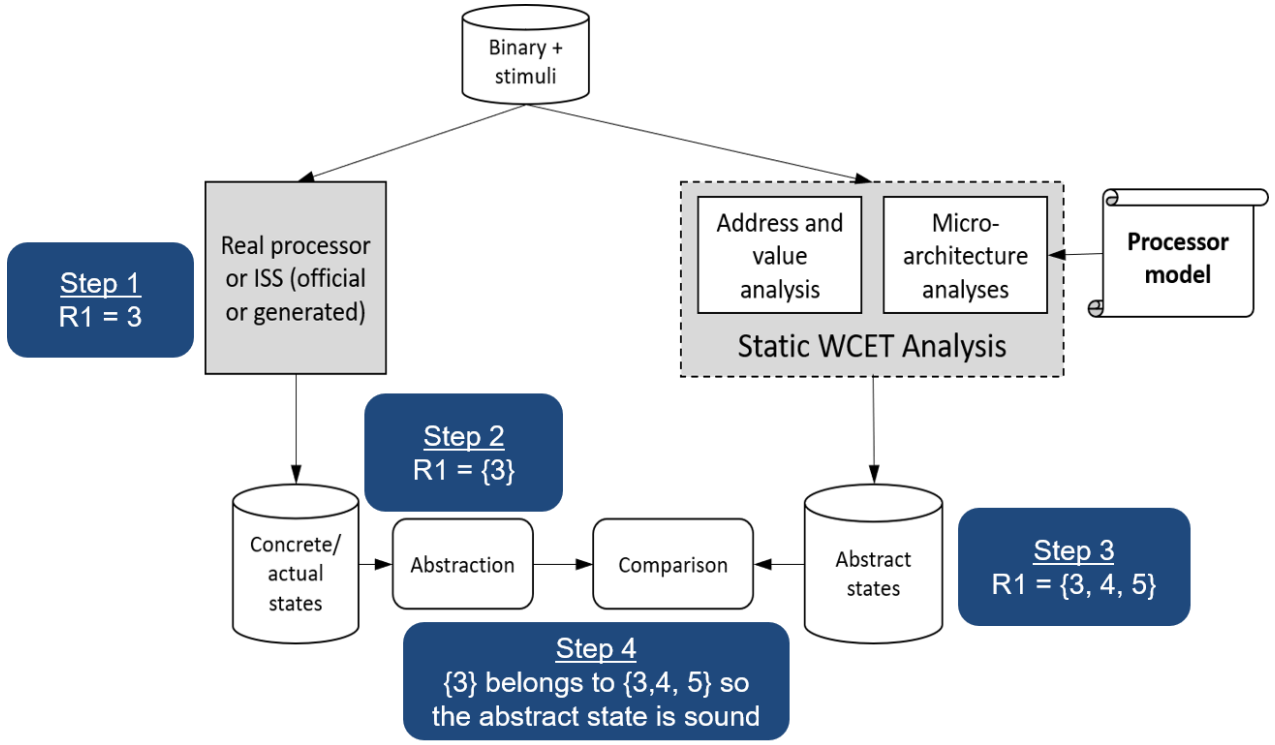
Abstract state: covers the concrete state

- **Checks for two things:**
  - If the value/address analysis is correctly implemented
  - If the semantic instruction is correctly implemented for TriCore

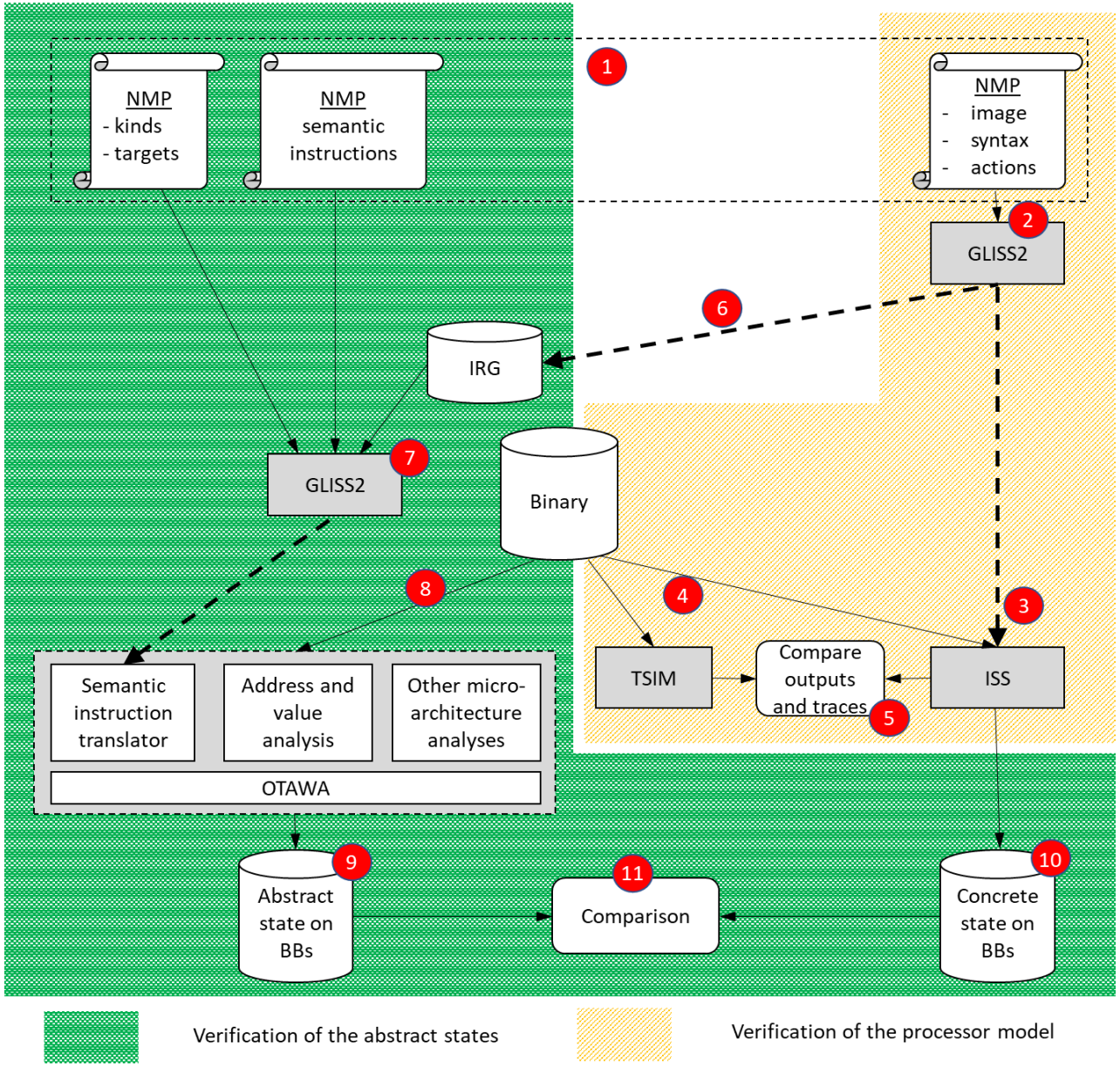


Some results

- **Correct some operations in the abstraction**
  - Validation of the tool is also important to have sound results
- **Some aspects were not taken into account in the previous TriCore implementation**
  - So far, we are contributing significantly for the TriCore support in OTAWA



Overall validation procedures



Future works

Next steps

- **Finish what we did**
  - Complete support of TriCore for OTAWA
- **Linkage to other our works**
  - Already have support for RISC-V
  - Extended to support FlexPRET
- **Possible track**
  - Generate semantic instructions from the ISA descriptions

Questions?